# **Org Chart Generator — Plain-English Documentation**

This guide explains what the app does and how each part works, in friendly language for someone with little or no coding background. It’s split into two pieces:

1. the **desktop app** (what you click and type in), and
2. the **engine** that loads data and draws the org chart image.

At the end, there’s a short section on installing the required software.

## **What the app does, end-to-end**

1. **You open the desktop app** and type a **Manager Name** and **Month and Year**. You can optionally tick boxes to show extra details (Location, Level).
2. **Choose your data source**:  
   * **Current data from a Workday Excel file on your computer** (click **Browse** to pick it).
   * **Past data from SharePoint** (tick **Past Data** and you’ll be asked to log in with your TD email and password). The file picker is disabled in this mode because data is pulled directly from SharePoint.
3. **Click “Generate Graph.”**
   * If you chose a local file, the app checks the file exists and then processes it.
   * If you chose past data, the app opens a small login window and, after you enter credentials, fetches the data from SharePoint.
4. Behind the scenes, the **engine**:  
   * Loads and cleans the data.
   * Builds the reporting tree under the manager you chose.
   * Draws a neat org chart image (PNG).
   * Adds branding (logo and labels) and saves a final image.

You’ll see a “Success” message when the chart is ready.

## **Part A — The Desktop App (file: form.py)**

**Class: OrgChartApp** This is the “windowed” app you use.

### **1) \_\_init\_\_(self) — starting the app window**

* Creates the main window (title, size, color theme).
* Keeps a list of any pop-up windows so they can be closed cleanly later.
* Builds all the buttons and fields by calling setup\_ui().
* Hooks the window close button (the “X”) to a safe shutdown.

### **2) setup\_ui(self) — laying out the screen**

* Adds the title at the top.
* Adds two text boxes: **Manager Name** and **Month and Year**.
* Adds checkboxes: **Show Location**, **Show Level**, and **Past Data** (for SharePoint).
* Adds a file picker for **Workday File** (only used when **Past Data** is *not* checked).
* Adds a **Generate Graph** button.

**Note:** When you select a file, a small reminder appears (“Ensure selected excel file is not open.”). It hides again if you clear the file.

### **3) select\_file(self) — picking a Workday Excel file**

* Opens a file browser limited to .xlsx.
* Saves your choice and shows the “close the file first” reminder.

### **4) on\_past\_data\_change(self) — switching to “Past Data” mode**

* If “Past Data” is checked:  
  + Disables the **Show Location** checkbox (because the historical data source doesn’t include location).
  + Clears the file path and hides the file reminder.
  + Shows two notes explaining how past data works.
* If “Past Data” is unchecked: re-enables **Show Location** and shows the file reminder again if a file is selected.

### **5) show\_login\_dialog(self) — logging into SharePoint**

* Opens a small secure pop-up that asks for your **TD Email** and **Password**.
* Only opens one login window at a time (brings it to front if already open).
* If email/password are blank, shows an error; otherwise proceeds and calls generate\_with\_past\_data(...).
* The **Cancel** button closes the pop-up cleanly.

### **6) generate\_with\_past\_data(self, email, password) — generate using SharePoint**

* Reads your Manager and Month/Year from the main window.
* Calls the engine to **load data from SharePoint** with your credentials, **clean and prepare it**, **draw the chart**, and **finalize the image**.
* Shows a success or error message.

### **7) button\_function(self) — main “Generate Graph” button**

* If **Past Data** is on → ensures the two fields are filled → opens the login dialog.
* If **Past Data** is off → ensures a valid local file is selected and both fields are filled → calls the engine to load, clean, draw, and finalize.

### **8) close\_all\_windows(self) and on\_closing(self) — safe shutdown**

* Closes any open pop-ups, releases keyboard focus, then closes the main window.
* This prevents “ghost” windows or the app getting stuck.

### **9) run(self) — start the app**

* Starts the app’s event loop and guarantees cleanup happens even if something goes wrong.

## **Part B — The Chart Engine (file: org.py)**

This is the logic that reads Excel data, understands who reports to whom, and draws the image files.

### **1) High-level overview (docstring)**

The engine:

* Loads, cleans, and processes org data.
* Handles name collisions (people with identical names).
* Maps job titles to job grades/levels.
* Draws the org chart using **Graphviz** and saves it as a PNG.
* Adds branding/labels to the final image.

It can be used by the desktop app or by itself from the command line.

### **2) Key constants**

* A system **PATH** update ensures Graphviz is found.
* The **SharePoint** site URL and relative file path for the historical “GOBS Monthly HC” workbook.

### **3) standardize\_columns(df)**

Different sheets sometimes rename columns slightly (e.g., “JOB TITLE” vs “TITLE”). This function normalizes those names so later steps always find the right columns (MANAGER, FULL NAME, TITLE, GRADE, WORKER TYPE).

### **4) load\_data(manager\_name, month\_year, email=None, password=None, file\_path=None)**

Two ways to load:

* **From SharePoint** (for past data): logs in using your email/password, downloads the Excel file in memory, finds the correct header row, and reads the sheet for the specified month. Then it standardizes column names.
* **From a local Workday export** (for current data): reads the first sheet, renames “Line Detail 1/2” to “Title/Location”, cleans the “Reports To” values (removing any leading codes/underscores), uppercases the headers, and trims stray spaces.

Returns a uniform table ready for the next step.

### **5) save\_df(datasheet, manager\_name, month\_year, file\_source\_flag=True)**

Creates the **clean, final dataset** to draw from and saves it to all\_reporting/all reporting employees to {manager}.xlsx:

* Removes duplicate columns and keeps the main fields.
* Ensures the chosen **manager** appears at the top of the list and appends **all people who report to them** (directly or indirectly).
* Loads a **Title → Level** lookup table and adds a “LEVEL” column.
* Ensures every person gets a unique ID, even if two people share the same name (see identify\_duplicate\_employees).
* Writes the cleaned sheet for the chosen month into the output Excel file (creates or updates).

For **past/SharePoint** mode (file\_source\_flag=False) it adapts the columns accordingly and still writes the same clean output.

### **6) get\_all\_reporting\_employees(avp\_name, df, visited=None)**

Finds **everyone who reports (even indirectly)** to a given manager by repeatedly walking down the reporting chain. It avoids loops using a visited set.

### **7) identify\_duplicate\_employees(df)**

If two rows have the same **FULL NAME**, the function makes the ID unique. If duplicates share the same manager, it adds numbers “(1)”, “(2)”; otherwise it appends the manager’s last name to distinguish them. This prevents name collisions in the drawing step.

### **8) load\_title\_level\_dict()**

Reads a small Excel lookup file (data/resources/title\_level\_dict.xlsx) to map job titles to levels (e.g., “Analyst” → “G6”).

### **9) build\_dict(reports\_by\_manager, root)**

Builds a nested structure of teams using a “depth-first search” (DFS). In plain terms: start at the root manager, list their direct reports, then each of those people’s reports, and so on.

### **10) generateGraph(df, manager\_name, month\_year\_value, show\_location=False, show\_level=True)**

Draws the org chart as a picture:

* Each person becomes a **box** with their **name** and **title**. You can optionally include their **level** and **location**, controlled by the checkboxes in the app.
* If a sub-team has **3 or fewer** people, they’re laid out **horizontally** under their manager.
* If a sub-team has **4 or more**, it creates a **vertical “bus” line** down the page and connects each person to it. This keeps big teams readable and compact.
* Saves a PNG like "ManagerName MonthYear Graph.png" into the raw\_graph folder and returns the base name for the next step.

### **11) finalize\_graph(org\_chart\_name, month\_year)**

Opens the raw chart image, places a **TD logo** and **department labels** at the top, adds extra white margins around the drawing for a clean page, writes the **Month Year** label, and saves the final image to the org chart folder (e.g., "ManagerName MonthYear Org Chart.png").

### **12) Command-line helpers — parse\_args() and main()**

If you ever want to run this **without** the desktop app (from a terminal), you can call:

python generateGraph.py "Manager Name" "May 2025"

The main() function wires together **load → save → draw → finalize** for that case.

## **Visual: simple workflow**

[You fill the form] → [Choose Data Source]

↓ ↓

Local Workday Excel SharePoint (Past Data)

↓ ↓

→→→ Load & Clean → Build Team Tree → Draw Chart → Add Branding → Save PNG

## 

## **Installation & Requirements**

All needed packages are listed in requirements.txt. The important ones are:

* **customtkinter** (desktop UI)
* **pandas, openpyxl** (Excel reading/cleaning)
* **graphviz** (drawing the org chart)
* **Pillow** (image/branding)
* **Office365-REST-Python-Client** (SharePoint access)

**Heads-up on typos to fix in requirements.txt:**

* openpyx]==3.1.5 should almost certainly be openpyxl==3.1.5.
* ur1lib3==2.5.0 should be urllib3==2.5.0.  
   Fixing these makes installation smoother.

**Graphviz system install:** Besides the Python package, Graphviz must also be installed on your machine so the dot program exists. The code sets a Windows PATH to a Graphviz folder so the engine can find it. If your Graphviz is elsewhere, update that PATH line.

## **Where files go**

* Cleaned, consolidated data: all\_reporting/all reporting employees to {Manager}.xlsx (multiple sheets inside, one per month).
* Raw chart images (pre-branding): raw\_graph/{Manager MonthYear} Graph.png
* Final chart images (with branding): org chart/{Manager MonthYear} Org Chart.png

## **Common questions**

**Q: Why does “Show Location” turn off in Past Data mode?** A: The historical SharePoint workbook doesn’t include that field in a consistent way, so the app disables it automatically when you pick Past Data.

**Q: What if two people have the same name?** A: The engine adds a small identifier (like “(1)”, “(2)”, or the manager’s last name) so each box is unique and the chart draws correctly.

**Q: My team is big and the chart gets messy — how is that handled?** A: For 4+ direct reports under the same manager, the engine switches to a vertical “bus” layout to keep things compact and readable.

## **Quick “How to Use” (non-technical)**

1. Open the app.
2. Enter **Manager Name** and **Month and Year** (e.g., May 2025).
3. Choose **Past Data** (SharePoint) *or* **Browse** for your local Workday Excel file.
4. (Optional) Tick **Show Level** / **Show Location** (location is disabled for Past Data).
5. Click **Generate Graph**.
6. Find your image in the **“org chart”** folder.